Homework 2 - Conditionals

CS 1301 - Intro to Computing - Spring 2022

Important

- Due Date: **Tuesday, January 25th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use Ed Discussion Forum
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this homework is for you to practice and understand how to write functions that implement conditionals. The homework will consist of 5 functions for you to implement. You have been given HW02.py skeleton file to fill out. Please read this PDF thoroughly as you will find more detailed information to complete your assignment.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Assata Quinichett (aquinichett@gatech.edu) and Josh Tabb (jtabb6@gatech.edu).

Helpful Information To Know

Print vs Return

Two concepts that may be difficult for beginner programmers to differentiate between are the print function and the return statement. While it may appear that they do the same thing, it is important to note that they are quite different. The purpose of the print function is to display information to the user. You cannot save what you print. The return statement, on the other hand, is part of a function definition. All functions have a return value, whether you explicitly write a return statement or not. Functions that do not explicitly have a return statement always return the value None. The return statement is useful because it allows you to give a value to a function. This allows you to either save it for later, use it in a more complex expression, or print it for human consumption.

For example, let's say we have the following two functions below in a file called `file.py`:

```
def printFunc():
    print(2)

def returnFunc():
    return 2
```

This is what would happen if we ran the file and typed the following into the Python shell:

```
>>> a = printFunc()
2
>>> print(a)
None
```

Notice that although the number '2' is printed to the screen, the variable we assigned the function call to, a, has the value None (`NoneType`).

```
>>> b = returnFunc()
>>> print(b)
2
```

When we call returnFunc() and assign it to a variable, nothing is printed to the screen because there are no print statements inside the function. However the variable, b, now holds the value 2 (`int`).

Workout

Function Name: workout()
Parameters: exerciseName (`str`), interestedFriends (`int`), totalFriends (`int`)
Returns: None (`NoneType`)
Description: This semester you've made it a goal to work out more often, but you don't want to do it alone. Write a function takes in the name of the exercise, the number of friends interested in that exercise, and the total number of friends you have invited.

If less than 20% of the friends you've invited want to exercise with you, print out a string in the following format:

```
"Let's try a different workout."
```

If 20% or more but less than 70% of your invited friends would like to exercise with you, print out a string in the following format:

```
"We will try to {exerciseName} for 30 minutes."
```

If 70% or more of your invited friends will exercise with you, print out a string in the following format:

```
"We are so excited to {exerciseName}!"
```

```
>>> workout("hike", 3, 6)
'We will try to hike for 30 minutes.'
```

```
>>> workout("dance", 7, 10)
'We are so excited to dance!'
```

Ice Cream

Function Name: iceCream()
Parameters: rating (`float`), distance (`float`)
Returns: choice (`str`)
Description: You and your friends had so much fun working out, and now you want to treat yourselves to ice cream! Using the table below, write a function that will help you decide which ice cream place to go to. You can only get ice cream from places that are above your desired rating and match your desired distance. Return the name of the ice cream place that matches your requirements. If there are no options that meet your requirements, then return “Try again tomorrow.”

Place	Rating	Distance
`"Jeni's"`	4.5	7.5
`"Cold Stone"`	4.5	3.6
`"Morelli's"`	4.5	4.2
`"Bruster's"`	4.0	1.3
`"Sweet Stack"`	4.0	6.4
`"Baskin Robbins"`	3.5	2.8

```
>>> iceCream(4.5, 3.6)
"Cold Stone"
```

```
>>> iceCream(3.5, 1.3)
"Try again tomorrow."
```

Restaurant Decider

Function Name: restaurantDecider()
Parameters: veganFriendly (`bool`), yelpStars (`int`), milesAway (`int`)
Returns: decisionStr (`str`)
Description: You and your friends decide to go out for dinner. But first, you need to pick out what restaurants will be good for your friend group. Write a function that takes in a `bool` representing whether or not a restaurant is vegan friendly, an `int` representing the number of yelp stars, and an `int` representing the distance of the restaurant.

- If a restaurant is not vegan friendly, then you should return the string `"Not tonight."`.
- If the restaurant is vegan friendly, but does not have at least 3 stars on yelp, then you should return the string `"Not good enough food."`.
- If the restaurant is vegan friendly, has at least 3 stars, but is more than 5 miles away, then return the string `"Too far!"`.
- If the restaurant is vegan friendly, has at least 3 stars, and is less than or equal to 5 miles away, then return the string `"Perfect restaurant!"`.

```
>>> restaurantDecider(True, 4, 7)
'Too far!'
```

```
>>> restaurantDecider(True, 2, 5)
'Not good enough food.'
```

Dinner Tip

Function Name: dinnerTip()
Parameters: numFriends (`int`), dinnerCost (`float`)
Returns: tipAmount (`float`)
Description: After a delicious meal, you face the dilemma of having to calculate the tip. Luckily, you have a formula to decide the amount you are going to tip. Write a function that takes in the number of friends and the total cost of the dinner and returns the tip amount. If the number of friends at dinner is less than or equal to 3, then you will tip 15% of the dinner cost. If the number of friends is greater than 3, but less than or equal to 7, then you will tip 20% of the dinner cost. If the number of friends is greater than 7, then you will tip 25% of the dinner cost. The tip amount should be **rounded to 2 decimal places**.

```
>>> dinnerTip(2, 10.0)
1.5
```

```
>>> dinnerTip(5, 35.0)
7.0
```

Plan Maker

Function Name: planMaker()
Parameters: timeA (`float`), costA (`int`), timeB (`float`), costB (`int`)
Returns: planDecision (`str`)
Description: You want to make plans with some of your friends, but being a very busy Georgia Tech student, you want to choose the plan that works best for you. Write a function that takes in the time and cost of two different plans and returns the cheapest or least time taking plan.

For this problem **cost takes precedence**. This means that you should first check the costs of both plans and return the plan that costs the least. If the plans cost the same you should return the plan that takes the least amount of time. If the cost and the time of the plans are the same, return `"No plans this weekend."`

```
>>> planMaker(5.5, 1000, 5.4, 5000)
"planA"
```

```
>>> planMaker(6.0, 10000, 6.0, 10000)
"No plans this weekend."
```

Grading Rubric

Function	Points
workout()	20
iceCream()	20
restaurantDecider()	20
dinnerTip()	20
planMaker()	20
Total	**100**

Provided

The `HW02.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW02.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the “Resubmit” button at the lower right-hand corner of Gradescope. You do not need to submit your `HW02.py` on Canvas.